UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/632,214	07/31/2003	Gerard Chauvel	TI-35427	1113

23494 7590 07/18/2008

TEXAS INSTRUMENTS INCORPORATED
P O BOX 655474, M/S 3999
DALLAS, TX 75265

EXAMINER	
PETRANEK, JACOB ANDREW	
ART UNIT	PAPER NUMBER
2183	

NOTIFICATION DATE	DELIVERY MODE
07/18/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

uspto@ti.com
uspto@dlemail.itg.ti.com

Office Action Summary	Application No.	Applicant(s)	
	10/632,214	CHAUVEL ET AL.	
	Examiner	Art Unit	
	Jacob Petranek	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 July 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3,5,8-11,13,15 and 18-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3,5,8-11,13,15 and 18-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *7/7/2008*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1, 3, 5, 8-11, 13, 15, and 18-22 are pending.

2. The office acknowledges the following papers:

IDS filed on 7/7/2008.

Claims and arguments filed on 4/25/2008.

IDS

3. The information disclosure statement filed 7/7/2008 fails to comply with 37 CFR 1.98(a)(2), which requires a legible copy of each cited foreign patent document; each non-patent literature publication or that portion which caused it to be listed; and all other information or that portion which caused it to be listed. It has been placed in the application file, but the information referred to therein has not been considered.

Maintained Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. §103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claim 9 is rejected under 35 U.S.C. §103(a) as being unpatentable over Terada et al. (U.S. 6,041,399), in view of Blaner et al. (U.S. 5,659,722), in view of Weaver et al. ("The SPARC Architecture Manual: Version 9").

6. As per claim 9:

Terada disclosed a method of executing an instruction defined by an opcode and having a reference to a register, an immediate value, and a control bit that dictates one of at least two tests, the method comprising:

Examining said control bits to determine its state (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(The opcode of the instruction specifies how the instruction will execute, which equates to the control bits.);

If said control bits are in a first state, comparing the immediate value to the contents of the register referenced in the instruction and skipping a subsequent instruction based on the outcome of the comparison (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(The compare instruction in figure 4 is done by comparing the data stored in the specified register to the immediate value. The register is a normal register within the register file and is not the status register. Thus having the same functionality.); or

Terada failed to teach if said control bit is in a second state, masking the contents of the register with the immediate value, testing one or more bits in the masked version of the contents of the register, and skipping a subsequent instruction based on the outcome of the testing; and wherein said control bit is outside said opcode.

However, Blaner disclosed if said control bits are in a second state, masking the contents of the register with the immediate value, testing one or more bits in the masked version of the contents of the register, and skipping a subsequent instruction based on the outcome of the testing (Blaner: Figures 3 and 7, column 4 lines 53-67 continued to column 5 lines 1-42 and column 6 lines 26-39 and column 8 lines 4-31)(The test and

branch instruction's opcode states that the comparison is done between the status register and the immediate predicate value within the instruction, which is element 508 in figure 3. Figure 7 shows the process of executing this instruction. Element 330 is the status register that stores the current status bits that are checked by the branch instruction. Column 8 discusses the process of masking the immediate mask in the instruction with the status register values to determine test value. This test value is the basis of if the branch is taken or not.).

Processing elements produce condition signals during execution that indicate conditions relevant to the execution of an instruction. These signals can later be used by conditional branches to determine if certain conditions were met for a branch instruction (Blaner: Column 1 lines 20-29). This type of branch instruction that performs a mask of the status bits with an immediate value to determine if the branch is taken would also be useful in other processors, such as the processor of Terada. The advantage of having conditional branches that branch on the status bits generated from prior instructions would have motivated one of ordinary skill in the art at the time of the invention to implement the test and branch instruction of Blaner into the processor of Terada. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to add the test and branch instruction to Terada for the benefits of branching dependent on different status bits generated from previous instructions.

Terada and Blaner failed to teach al control bit that specifies a first or second state and wherein said control bit is outside said opcode.

However, Weaver disclosed a control bit that specifies a first or second state and wherein said control bit is outside said opcode (Weaver: Figure 33, pages 64, 138-139, and 146-150)(The SPARC manual shows branch instructions with a common opcode and a rcond code that specifies a specific type of branch instruction. The combination with Terada and Blaner allows for the two instructions to share an opcode and have control bits outside of the opcode to specify which operation will execute. An individual bit within the rcond code allows for the selection of two operations given the other bits are the same value.).

The advantage of using control bits outside of an instruction opcode to specify execution is that is can be used to expand a processor's instruction set architecture for operations that have unused bits in their instruction encoding. One of ordinary skill in the art would have been motivated by this to allow for control bits outside of the instruction opcode to specify different operations for the advantage of expanding a processor's instruction set architecture. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement control bits outside of the instruction opcode for the advantage of expanding a processor's instruction set architecture.

7. Claim 10 is rejected under 35 U.S.C. §103(a) as being unpatentable over Terada et al. (U.S. 6,041,399), in view of Blaner et al. (U.S. 5,659,722), in view of Weaver et al. ("The SPARC Architecture Manual: Version 9"), further in view of Chen et al. (U.S. 5,504,903)

8. As per claim 10:

Terada, Blaner, and Weaver disclosed the method of claim 9.

Terada, Blaner, and Weaver failed to teach wherein skipping the subsequent instruction comprises replacing the subsequent instruction with a no operation instruction.

However, Chen disclosed wherein skipping the subsequent instruction comprises replacing the subsequent instruction with a no operation instruction (Chen: Column 7 lines 59-67 continued to column 8 lines 1-3).

Both the bit test and skip if set/clear instructions are essentially a predicated compare instruction, which will only execute the next instruction if a condition is met. If the condition is met, then the next instruction is not allowed to complete and is essentially the same as a nop instruction. Thus, it would have been obvious to one of ordinary skill in the art to use the process from Chen of substituting in a nop instruction instead of the instruction from Blaner or Terada that will simply not complete if the condition to not execute is met.

9. Claims 1, 3, 5, 8, and 21 are rejected under 35 U.S.C. §103(a) as being unpatentable over Terada et al. (U.S. 6,041,399), in view of Ramasamy et al. (U.S. 6,931,632), in view of Blaner et al. (U.S. 5,659,722), in view of Weaver et al. ("The SPARC Architecture Manual: Version 9").

10. As per claim 1:

Terada disclosed a processor executing a plurality of instructions, comprising:

An arithmetic logic unit (Terada: Figure 1 elements 103-104 and 203-204, column 5 lines 35-51); and

A plurality of registers coupled to the ALU, each register programmable to store a register value (Terada: Figure 1 elements 102 and 202, column 5 lines 52-61);

Wherein said processor executes a routine having a test and skip instruction defined by an opcode, said instruction includes an immediate value and a reference to a register, the test and skip instruction performs a comparison using the immediate value and the register value stored in the referenced register, and selectively skips a subsequent instruction that follows the test and skip instruction based on the comparison (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(It's obvious to one of ordinary skill in the art that the instructions contained within figure 4 could occur within a routine. The compare greater than instruction in figure 4 compares a register value to an immediate value. The predicate result of this will cause the subtraction instruction to execute or execute and not save the results of the instruction to a register. Thus having the same functionality.).

Wherein the test and skip instruction includes at least one bit that specifies whether the register reference is to a register from a first group of registers or to a register from a second group of registers (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(The first group of registers is registers from the register file and the second group of registers is the status register. The compare instruction's opcode specifies that a comparison will be done by a normal register and not a status register. Thus having the same functionality.); and if a register from the first

group of registers is specified by said at least one bit, the comparison is performed by comparing the immediate value to the register value (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(The compare instruction in figure 4 is done by comparing the data stored in the register to the immediate value. Thus having the same functionality.).

Terada failed to teach if a register from the second group of registers is specified by said at least one bit, the comparison is performed by masking the register value with the immediate value and examining one or more bits in the masked version of the referenced register and wherein the subsequent instruction jumps to another routine.

However, Ramasamy disclosed wherein the subsequent instruction jumps to another routine (Ramasamy: Column 1 lines 38-45)(Ramasamy disclosed a predicated branch instruction that relies on a predicated value to determine if the branch instruction is taken or not. Since the branch is a call instruction, it jumps to another routine.).

The use of a predicated branch instruction has the same result of a conditional branch instruction, which both result in branching to the target address if a particular condition is met. Thus, it's obvious to one of ordinary skill in the art at the time of the invention that each of these methods could be used intertwined to achieve the same result.

Terada and Ramasamy failed to teach if a register from the second group of registers is specified by said at least one bit, the comparison is performed by masking the register value with the immediate value and examining one or more bits in the masked version of the referenced register.

However, Blaner disclosed if a register from the second group of registers is specified by said bit, the comparison is performed by masking the register value with the immediate value and examining one or more bits in the masked version of the referenced register (Blaner: Figures 3 and 7, column 4 lines 53-67 continued to column 5 lines 1-42 and column 6 lines 26-39 and column 8 lines 4-31)(The test and branch instruction's opcode states that the comparison is done between the status register and the immediate predicate value within the instruction, which is element 508 in figure 3. Figure 7 shows the process of executing this instruction. Element 330 is the status register that stores the current status bits that are checked by the branch instruction. Column 8 discusses the process of masking the immediate mask in the instruction with the status register values to determine test value. This test value is the basis of if the branch is taken or not.).

Processing elements produce condition signals during execution that indicate conditions relevant to the execution of an instruction. These signals can later be used by conditional branches to determine if certain conditions were met for a branch instruction (Blaner: Column 1 lines 20-29). This type of branch instruction that performs a mask of the status bits with an immediate value to determine if the branch is taken would also be useful in other processors, such as the processor of Terada. The advantage of having conditional branches that branch on the status bits generated from prior instructions would have motivated one of ordinary skill in the art at the time of the invention to implement the test and branch instruction of Blaner into the processor of Terada. Thus, it would have been obvious to one of ordinary skill in the art at the time

of the invention to add the test and branch instruction to Terada for the benefits of branching dependent on different status bits generated from previous instructions.

Terada, Ramasamy, and Blaner failed to teach an instruction with a control bit to control two different modes of comparison and wherein the test and skip instruction includes at least one bit, separate from the opcode.

However, Weaver disclosed an instruction with a control bit to control two different modes of comparison and wherein the test and skip instruction includes at least one bit, separate from the opcode (Weaver: Figure 33, pages 64, 138-139, and 146-150)(The SPARC manual shows branch instructions with a common opcode and a rcond code that specifies a specific type of branch instruction. The combination with Terada and Blaner allows for the two instructions to share an opcode and have control bits outside of the opcode to specify which operation will execute. An individual bit within the rcond code allows for the selection of two operations given the other bits are the same value.).

The advantage of using control bits outside of an instruction opcode to specify execution is that is can be used to expand a processor's instruction set architecture for operations that have unused bits in their instruction encoding. One of ordinary skill in the art would have been motivated by this to allow for control bits outside of the instruction opcode to specify different operations for the advantage of expanding a processor's instruction set architecture. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement control bits outside of the instruction opcode for the advantage of expanding a processor's instruction set

architecture.

11. As per claim 3:

Terada, Ramasamy, Blaner, and Weaver disclosed the processor of claim 1 wherein, if comparing the immediate value to the register value, the processor skips the subsequent instruction if the immediate value does not match the register value and executes the subsequent if the immediate value does match the register value (Terada: Figure 11, column 5 lines 66-67 continued to column 6 lines 1-24 and column 7 lines 39-48)(Figure 11 shows a compare equal instruction that compares a immediate value to a register value. The following instruction in figure 11 is skipped if the results aren't equal and is executed if they are equal. Thus having the same functionality.)

12. As per claim 5:

Terada, Ramasamy, Blaner, and Weaver disclosed the processor of claim 1 wherein, if masking the register value, the masking is performed by ANDing the immediate value with the register value (Blaner: Figure 7 elements 352, 362, 372, and 382, column 8 lines 4-31)(These elements are ANDing the value from the status register with the immediate value of the predicate from the branch instruction.).

13. As per claim 8:

Terada, Ramasamy, Blaner, and Weaver disclosed the processor of claim 1

Wherein the registers include a status register (Blaner: Figure 5, column 7 lines 1-33)(The multiple predicate register stores predicate value for instructions that don't have an immediate predicate value and also stores status bits for each processing element. Thus having the same functionality.); and

If the register reference specified by said at least one bit is not the status register, the comparison is performed by comparing the immediate value to the register value in the referenced register (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(The compare instruction in figure 4 is done by comparing the data stored in the specified register to the immediate value. The register is a normal register within the register file and is not the status register. Thus having the same functionality.)

If the register reference specified by said at least one bit is the status register, the comparison is performed by masking the register value in the status register with the immediate value and examining one or more bits in the masked version of the status register (Blaner: Figures 3 and 7, column 4 lines 53-67 continued to column 5 lines 1-42 and column 6 lines 26-39 and column 8 lines 4-31)(The test and branch instruction's opcode states that the comparison is done between the status register and the immediate predicate value within the instruction, which is element 508 in figure 3. Figure 7 shows the process of executing this instruction. Element 330 is the status register that stores the current status bits that are checked by the branch instruction. Column 8 discusses the process of masking the immediate mask in the instruction with the status register values to determine test value. This test value is the basis of if the branch is taken or not.).

It would have been obvious to one of ordinary skill in the art that having an instruction with two different modes of operation using a control bit is essentially the same as having two different instructions using two separate opcodes to control two different modes of operation. Thus, it would have been obvious to one of ordinary skill

in the art at the time of the invention to combine the two instructions and use a single control bit in the opcode to differentiate between the two modes of execution. In addition, according to "In re Larson" (144 USPQ 374 (CCPA 1965)), to make integral doesn't give patentability over prior art.).

14. As per claim 21:

Terada, Ramasamy, Blaner, and Weaver disclosed the processor of claim 1 wherein, if masking the register value, the processor skips the subsequent instruction if the masked version of the register value comprises all logic high values or all logic low values, and executes the subsequent instruction if the masked version comprises a mix of logic high and low values (Blaner: Figures 3 and 7, column 4 lines 53-67 continued to column 5 lines 1-42, column 6 lines 26-39, and column 8 lines 4-31)(Column 8 discusses the process of masking the immediate mask in the instruction with the status register values to determine test value. This test value is the basis of if the branch is taken or not. At line 25, a branch occurs if all values are high. Blaner only discusses using an AND or an OR circuit to determine branching based on the masked values. One of ordinary skill in the art would realize that other logic gates can be useful for masking values to get a certain result, such as NOT, XOR, and XNOR gates. An XNOR gate will output a 1 value if all values are either 0 or 1 and will output a 0 value if there's a mix of 0's and 1's. It's obvious to one of ordinary skill in the art that these gates are able to be substituted to get to a specific solution. Thus, it's obvious to one of ordinary skill in the art to substitute the XNOR gate for the AND and OR gates used within Blaner.).

15. Claims 11, 13, 15, 18, and 22 are rejected under 35 U.S.C. §103(a) as being unpatentable over Terada et al. (U.S. 6,041,399), in view of Feierbach et al. (U.S. 6,088,786), in view of Blaner et al. (U.S. 5,659,722), in view of Weaver et al. ("The SPARC Architecture Manual: Version 9").

16. As per claim 11:

Terada disclosed a system, comprising:

A main processor unit (Terada: Figure 16 figure 20, column 10 lines 43-49); and

A co-processor coupled to said main processor unit (Terada: Figure 16 element 22, column 10 lines 43-49);

Wherein, during the register-based instruction mode, the coprocessor executes an instruction defined by an opcode and including an immediate value and a reference to a register accessible to said co-processor, performs a comparison using the immediate value and the register value, and executes or skips a subsequent instruction based on the comparison (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(The compare greater than instruction in figure 4 compares a register value to an immediate value. The predicate result of this will cause the subtraction instruction to execute or execute and not save the results of the instruction to a register. Thus having the same functionality. Since the instruction deals with registers, the coprocessor executes the comparison instruction in a register-based mode.).

Wherein the instruction includes at least one bit that specifies whether the register reference is to a register from a first group of registers or to a register from a

second group of registers (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(The first group of registers is registers from the register file and the second group of registers is the status register. The compare instruction's opcode specifies that a comparison will be done by a normal register and not a status register. Thus having the same functionality.) and if a register from the first group of registers is specified by said at least one bit, the comparison is performed by comparing the immediate value to the register value (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(The compare instruction in figure 4 is done by comparing the data stored in the register to the immediate value. Thus having the same functionality.).

Terada failed to teach if a register from the second group of registers is specified by said at least one bit, the comparison is performed by masking the register value with the immediate value and examining one or more bits in the masked version of the referenced register and wherein said co-processor selectively operates in a stack-based instruction mode and a register-based instruction mode.

However, Feierbach disclosed wherein said co-processor selectively operates in a stack-based instruction mode and a register-based instruction mode (Feierbach: Figure 2 element 227, column 7 lines 27-37)(Figure 2 shows a processor that is able to selectively execute stack-based instructions and register-based instructions by using a predecoder, element 227, to determine where the current instruction is to go.).

The advantage of stack-based processors is that they are much more compact and efficient than there register-based counterparts. Having both a stack-based and register-based processor is advantageous when a processor also has to occasionally

execute high-performance multimedia applications, which are better suited for register-based processors (Feierbach: Column 2 lines 44-67 continued to column 3 lines 1-45). One of ordinary skill in the art would have been motivated by the increased performance in certain applications for stack-based processors to add a stack-based processor to the processor of Terada. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a stack-based processor alongside the register-based processor of Terada for the advantage of increased performance in certain applications.

Terada and Feierbach failed to teach if a register from the second group of registers is specified by said at least one bit, the comparison is performed by masking the register value with the immediate value and examining one or more bits in the masked version of the referenced register.

However, Blaner disclosed if a register from the second group of registers is specified by said bit, the comparison is performed by masking the register value with the immediate value and examining one or more bits in the masked version of the referenced register (Blaner: Figures 3 and 7, column 4 lines 53-67 continued to column 5 lines 1-42 and column 6 lines 26-39 and column 8 lines 4-31)(The test and branch instruction's opcode states that the comparison is done between the status register and the immediate predicate value within the instruction, which is element 508 in figure 3. Figure 7 shows the process of executing this instruction. Element 330 is the status register that stores the current status bits that are checked by the branch instruction. Column 8 discusses the process of masking the immediate mask in the instruction with

the status register values to determine test value. This test value is the basis of if the branch is taken or not.).

Processing elements produce condition signals during execution that indicate conditions relevant to the execution of an instruction. These signals can later be used by conditional branches to determine if certain conditions were met for a branch instruction (Blaner: Column 1 lines 20-29). This type of branch instruction that performs a mask of the status bits with an immediate value to determine if the branch is taken would also be useful in other processors, such as the processor of Terada. The advantage of having conditional branches that branch on the status bits generated from prior instructions would have motivated one of ordinary skill in the art at the time of the invention to implement the test and branch instruction of Blaner into the processor of Terada. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to add the test and branch instruction to Terada for the benefits of branching dependent on different status bits generated from previous instructions.

Terada, Feierbach, and Blaner failed to teach an instruction with a control bit to control two different modes of comparison and wherein the instruction includes at least one bit, separate from said opcode.

However, Weaver disclosed an instruction with a control bit to control two different modes of comparison and wherein the instruction includes at least one bit, separate from said opcode (Weaver: Figure 33, pages 64, 138-139, and 146-150)(The SPARC manual shows branch instructions with a common opcode and a rcond code that specifies a specific type of branch instruction. The combination with Terada and

Blaner allows for the two instructions to share an opcode and have control bits outside of the opcode to specify which operation will execute. An individual bit within the rcond code allows for the selection of two operations given the other bits are the same value.).

The advantage of using control bits outside of an instruction opcode to specify execution is that is can be used to expand a processor's instruction set architecture for operations that have unused bits in their instruction encoding. One of ordinary skill in the art would have been motivated by this to allow for control bits outside of the instruction opcode to specify different operations for the advantage of expanding a processor's instruction set architecture. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement control bits outside of the instruction opcode for the advantage of expanding a processor's instruction set architecture.

17. As per claim 13:

Claim 13 essentially recites the same limitations of claim 3. Therefore, claim 13 is rejected for the same reasons as claim 3.

18. As per claim 15:

Claim 15 essentially recites the same limitations of claim 5. Therefore, claim 15 is rejected for the same reasons as claim 5.

19. As per claim 18:

Terada, Feierbach, Blaner, and Weaver disclosed the system of claim 11 further comprising wireless communication circuitry and said system comprises a cell phone

(Official notice is taken that the processing system could be part of a cellular telephone.).

20. As per claim 22:

Claim 22 essentially recites the same limitations of claim 21. Therefore, claim 22 is rejected for the same reasons as claim 21.

21. Claims 19-20 are rejected under 35 U.S.C. §103(a) as being unpatentable over Terada et al. (U.S. 6,041,399), in view of Ramasamy et al. (U.S. 6,931,632), in view of Hammond et al. (U.S. 5,638,525), in view of Blaner et al. (U.S. 5,659,722), in view of Weaver et al. ("The SPARC Architecture Manual: Version 9").

22. As per claim 19:

Terada disclosed a programmable logic device comprising:

Control logic (Terada: Figures 1 and 5, columns 5-6); and

Means for decoding a control bit in an instruction that includes an immediate value and a reference to a register for performing a comparison using the immediate value and a register value stored in the referenced register, and for causing the processor to execute or skip a subsequent instruction that follows the instruction based on the comparison (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(The compare greater than instruction in figure 4 compares a register value to an immediate value. The predicate result of this will cause the subtraction instruction to execute or execute and not save the results of the instruction to a register. Thus having the same functionality. The compare instruction's opcode contains control bits that

specifies a comparison will be done by a normal register and not a status register.).

Wherein said control bit selectively specifies whether the comparison is to be performed by comparing the immediate value to the register value (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(The first group of registers is registers from the register file and the second group of registers is the status register. The compare instruction's opcode specifies that a comparison will be done by a normal register and not a status register. Thus having the same functionality.).

Terada failed to teach means for selectively changing an operating mode of the programmable logic device; whether the comparison is to be performed by masking the register value with the immediate value and examining one or more bits in the masked version of the referenced register; and wherein the subsequent instruction jumps to a routine associated with a particular operating mode.

However, Ramasamy disclosed wherein the subsequent instruction jumps to a routine (Ramasamy: Column 1 lines 38-45)(Ramasamy disclosed a predicated branch instruction that relies on a predicated value to determine if the branch instruction is taken or not. Since the branch is a call instruction, it jumps to another routine.).

The use of a predicated branch instruction has the same result of a conditional branch instruction, which both result in branching to the target address if a particular condition is met. Thus, it's obvious to one of ordinary skill in the art at the time of the invention that each of these methods could be used intertwined to achieve the same result.

Terada and Ramasamy failed to teach means for selectively changing an

operating mode of the programmable logic device and whether the comparison is to be performed by masking the register value with the immediate value and examining one or more bits in the masked version of the referenced register.

However, Hammond disclosed means for selectively changing an operating mode of the programmable logic device (Hammond: Figure 2 element 212, column 4 lines 61-67 continued to column 5 lines 1-19)(The switch instruction changes the processor from one operating mode to the other.); and

Wherein the subsequent instruction jumps to a routine associated with a particular operating mode (Hammond: Figure 2 element 212, column 4 lines 61-67 continued to column 5 lines 1-19)(Ramasamy: Column 1 lines 38-45)(The combination of Hammond and Ramasamy result in the jump switch instruction of Hammond being a predicated branch instruction.).

The advantage of using multiple instruction sets for a processor is that it allows for increased flexibility in what type of programs the processor can execute. One of ordinary skill in the art would have been motivated by this to implement a processor with multiple ISA's. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a processor with multiple instruction set architectures for the advantage of increased flexibility in the type of programs that the processor is able to execute.

Terada, Ramasamy, and Hammond failed to teach whether the comparison is to be performed by masking the register value with the immediate value and examining one or more bits in the masked version of the referenced register.

However, Blaner disclosed whether the comparison is to be performed by masking the register value with the immediate value and examining one or more bits in the masked version of the referenced register (Blaner: Figures 3 and 7, column 4 lines 53-67 continued to column 5 lines 1-42 and column 6 lines 26-39 and column 8 lines 4-31)(The test and branch instruction's opcode states that the comparison is done between the status register and the immediate predicate value within the instruction, which is element 508 in figure 3. Figure 7 shows the process of executing this instruction. Element 330 is the status register that stores the current status bits that are checked by the branch instruction. Column 8 discusses the process of masking the immediate mask in the instruction with the status register values to determine test value. This test value is the basis of if the branch is taken or not.).

Processing elements produce condition signals during execution that indicate conditions relevant to the execution of an instruction. These signals can later be used by conditional branches to determine if certain conditions were met for a branch instruction (Blaner: Column 1 lines 20-29). This type of branch instruction that performs a mask of the status bits with an immediate value to determine if the branch is taken would also be useful in other processors, such as the processor of Terada. The advantage of having conditional branches that branch on the status bits generated from prior instructions would have motivated one of ordinary skill in the art at the time of the invention to implement the test and branch instruction of Blaner into the processor of Terada. Thus, it would have been obvious to one of ordinary skill in the art at the time

of the invention to add the test and branch instruction to Terada for the benefits of branching dependent on different status bits generated from previous instructions.

Terada, Ramasamy, Hammond, and Blaner failed to teach an instruction with a control bit to control two different modes of comparison and wherein said control bit is separate from an opcode that defines said instruction.

However, Weaver disclosed an instruction with a control bit to control two different modes of comparison and wherein said control bit is separate from an opcode that defines said instruction (Weaver: Figure 33, pages 64, 138-139, and 146-150)(The SPARC manual shows branch instructions with a common opcode and a rcond code that specifies a specific type of branch instruction. The combination with Terada and Blaner allows for the two instructions to share an opcode and have control bits outside of the opcode to specify which operation will execute. An individual bit within the rcond code allows for the selection of two operations given the other bits are the same value.).

The advantage of using control bits outside of an instruction opcode to specify execution is that is can be used to expand a processor's instruction set architecture for operations that have unused bits in their instruction encoding. One of ordinary skill in the art would have been motivated by this to allow for control bits outside of the instruction opcode to specify different operations for the advantage of expanding a processor's instruction set architecture. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement control bits outside of the instruction opcode for the advantage of expanding a processor's instruction set architecture.

23. As per claim 20:

Terada, Ramasamy, Hammond, Blaner, and Weaver disclosed the system of claim 19 including means for comparing the immediate value to the register value in the referenced register (Terada: Figure 4, column 5 lines 66-67 continued to column 6 lines 1-24)(The compare greater than instruction in figure 4 compares a register value to an immediate value. The predicate result of this will cause the subtraction instruction to execute or execute and not save the results of the instruction to a register. Thus having the same functionality.).

Response to Arguments

24. The arguments presented by Applicant in the response, received on 4/25/2008 are not considered persuasive.

25. Applicant argues that "Claim 1 requires a single instruction capable of performing two different types of comparison operations based on the status of a bit which is not part of the opcode. The Examiner has been forced to reject claim 1 over the combination of no less than four references ... As the Examiner is no doubt aware, an obviousness rejection cannot be based on hindsight gleaned from the inventor's own teachings. Further, the CAFC has emphasized this point. "This is essential for combination inventions, for generally all combinations are of known elements." *Interconnect Planninq Corp. v. Feil,* 774 F.3d 1132, 1143 (Fed. Cir. 1985). Applicants submit that, absent the hindsight of Applicants' own specification, one of ordinary skill in

the art would not have been motivated to seek out and combine the four references used by the Examiner" for claim 1.

This argument is not found to be persuasive for the following reason.

In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

In response to applicant's argument that the examiner has combined an excessive number of references, reliance on a large number of references in a rejection does not, without more, weigh against the obviousness of the claimed invention. See *In re Gorman*, 933 F.2d 982, 18 USPQ2d 1885 (Fed. Cir. 1991).

As the applicant submits that the claim 1 "is a single instruction capable of performing two different types of comparison operations," it shouldn't be too surprising that more than one reference is used to reject the claim if a single instruction isn't found in the prior art to perform the two operations. The combination of two known instructions into a single instruction is an obvious modification to one of ordinary skill in the art for the advantage of freeing up valuable opcodes for executing other types of operations. This is further obvious to one of ordinary skill in the art to combine two

known operations into a single opcode because opcodes values are always in demand as processors add more and more types of operations into the processor. Thus, sufficient motivation is found for combining Terada, Blaner, and Weaver to teach the two different types of instructions into a single comparison instruction of Terada for the advantage of freeing an opcode value.

The last reference of Ramasamy disclosed the limitation of a predicated branch instruction. The use of a predicated branch instruction has the same result of a conditional branch instruction, which both result in branching to the target address if a particular condition is met. Thus, it's obvious to one of ordinary skill in the art at the time of the invention that each of these methods could be used intertwined to achieve the same result.

Therefore, sufficient motivation is given to combine the four reference to reject the claimed invention for claim 1.

The examiner notes that an amendment clarifying the control bit for selecting one of the two types of comparison operations would overcome all of the current rejections. Figure 5 shows that the control bit is the most significant bit of the source register of the instruction. If the control bit was claimed to be a bit within the source register of the instruction, then this would overcome the current rejections.

26. Applicant argues "Claims 21 and 22 require an assessment of whether the masked version of the register is all high/low values or a mix of high and low values. The Examiner used Blaner for this teaching, but acknowledged that Blaner does not specifically teach this limitation. The Examiner noted that Blaner teaches the use of

AND and OR gates, but concluded that it would have been obvious to use an exclusive NOR (XNOR) gate. An XNOR gate outputs a high if the input values are all 0 or all 1, otherwise a logic low is output (mix logic inputs). Applicants respectfully request the Examiner to justify why he believes this specific limitation would have been obvious even though the cited art admittedly does not teach it. Further, absent hindsight from Applicants' specification (which cannot be used)" for claims 21 and 22.

This argument is found to be persuasive for the following reason.

In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

Blaner disclosed many of the possible types of operations that could result in taking a branch based on the masking operation in column 8 lines 20-31. One of ordinary skill in the art would also realize that additional types of tests can be performed with other commonly-known types of logic gates, such as an XNOR gate. This allows the additional flexibility to branch on all bits being 1 or 0.

Conclusion

THIS ACTION IS MADE FINAL.

The following is text cited from 37 CFR 1.111(c): In amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jacob Petranek whose telephone number is 571-272-5988. The examiner can normally be reached on M-F 8:00-4:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Jacob Petranek
Examiner, Art Unit 2183

/Eddie P Chan/

Supervisory Patent Examiner, Art Unit 2183